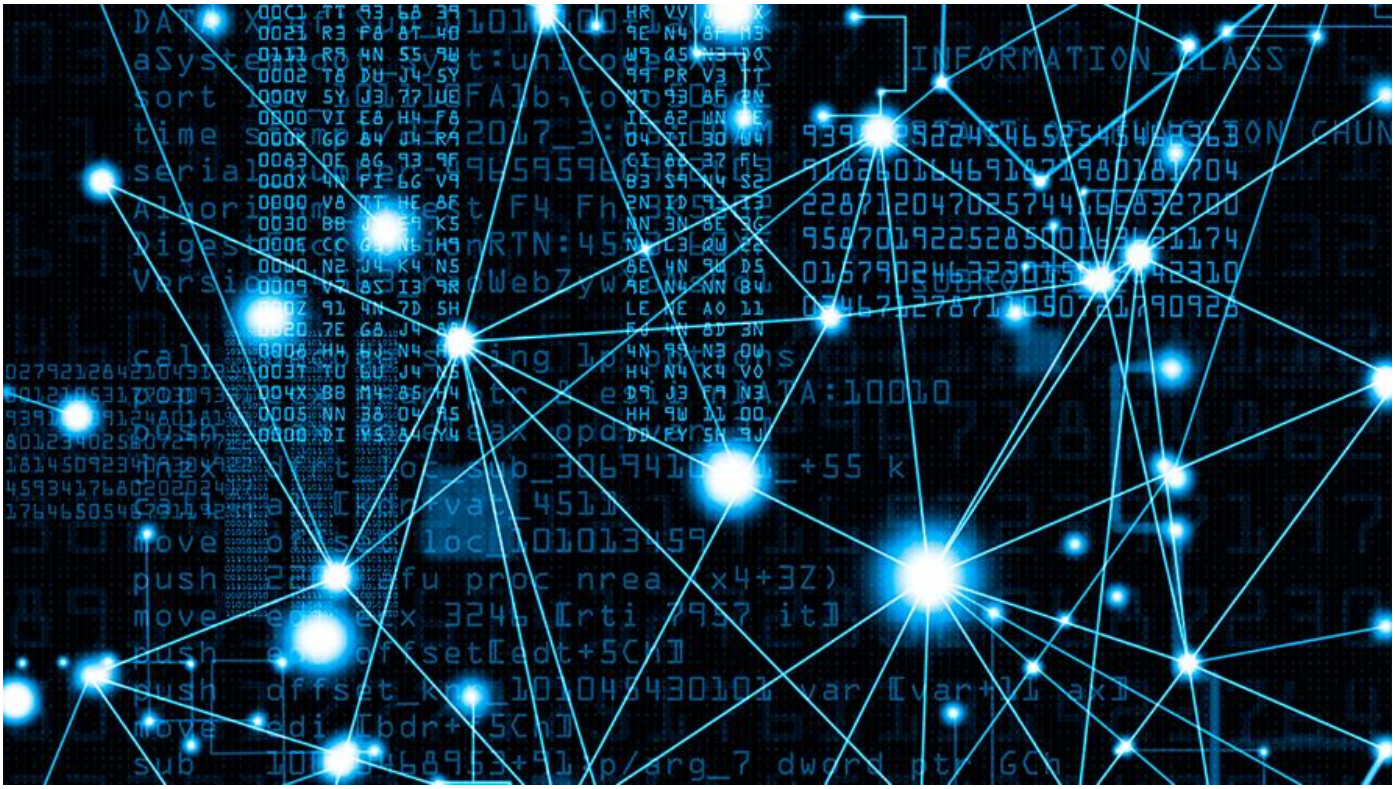
ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΠΟΛΥΠΛΟΚΟΤΗΤΑ

3Η ΣΕΙΡΑ ΓΡΑΠΤΩΝ ΑΣΚΗΣΕΩΝ



FEBRUARY 9, 2022

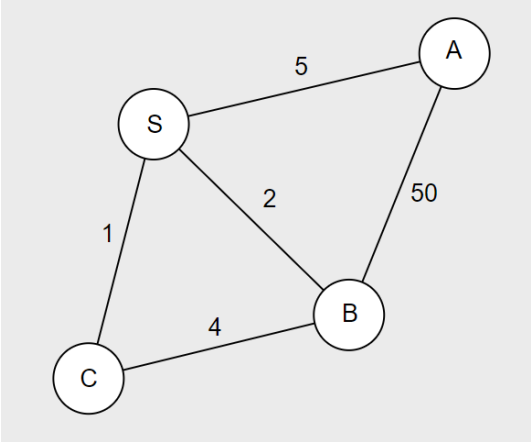
ΘΟΔΩΡΗΣ ΑΡΑΠΗΣ – EL18028

Άσκηση 1

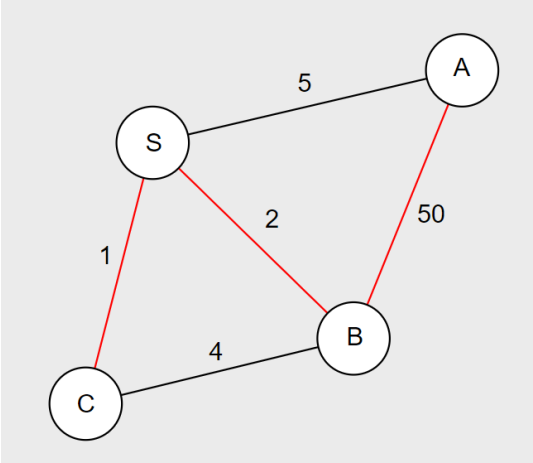
(α)

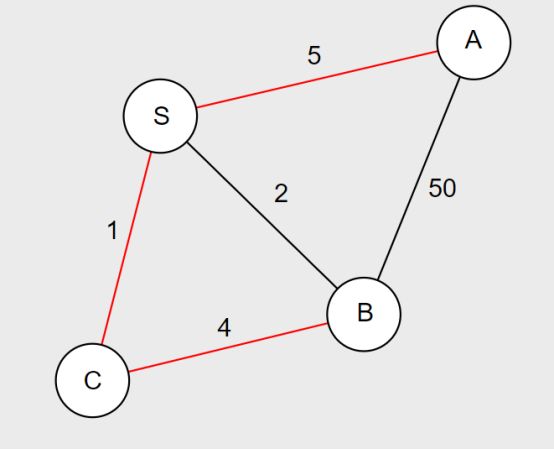
Μπορούμε να το δείξουμε εύκολα αυτό με ένα αντιπαράδειγμα:

Έχουμε το ακόλουθο γράφημα:



Για $k = 2$ ο άπληστος αλγόριθμος θα μας έδινε το ΕΣΔ (συνολικού βάρους 53) που φαίνεται στην αριστερή εικόνα (κόκκινες γραμμές) καθώς θα επέλεγε τις ακμές με τα μικρότερα βάρη που συνδέονται στον κόμβο S. Όμως το ΕΣΔ στο συγκεκριμένο γράφημα με τον κόμβο S να έχει βαθμό 2 φαίνεται στην δεξιά εικόνα και έχει βάρος 10.





(β)

Δίνεται γράφος $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{w})$

Αλγόριθμος

Έστω E_S το σύνολο των ακμών που προσπίπτουν στον κόμβο S. Δημιουργούμε μία λίστα L με όλες τις δυνατές διαφορές βαρών $w(e) - w(e_S), \forall e \in E \; και \; \forall e_S \in E_S$ και την ταξινομούμε $\left(O\left(|E||E_S|log(|E||E_S|)\right)\right)$ σε αύξουσα σειρά. Στη συνέχεια εκτελούμε δυαδική αναζήτηση στην λίστα αυτή $\left(O\left(log(|E||E_S|)\right)\right)$ και το στοιχείο $\ell \in L$ που λαμβάνουμε το προσθέτουμε σε κάθε ακμή του E_S. Ύστερα εκτελούμε τον αλγόριθμο Kruskal $\left(O\left(|E|log(|E|)\right)\right)$ (θεωρούμε ότι οι ακμές του E_S επιλέγονται με μεγαλύτερη προτεραιότητα από τον Kruskal) και ελέγχουμε τον βαθμό του κόμβου S. Αν είναι μικρότερος του k τότε αναζητούμε για μεγαλύτερα στοιχεία ℓ και αντιστρόφως, επαναλαμβάνοντας την παραπάνω διαδικασία. Αν είναι ίσος με το k τερματίζουμε. (Πιθανόν, όταν έχουμε περισσότερες από μία ακμές του E_S με το ίδιο βάρος, να χρειάζεται στην τελευταία επανάληψη της δυαδικής αναζήτησης (όταν $high = low$) να τροποποιήσουμε τον Kruskal ώστε να επιλέξει συνολικά k ακμές του E_S και να αγνοήσει τους περίσσιους).

Ορθότητα

Αρχικά, εύκολα μπορούμε να αντιληφθούμε ότι για μικρά ℓ ο βαθμός του κόμβου S θα αυξάνεται, ενώ αντίθετα για μεγάλα ℓ ο βαθμός του θα μειώνεται. Αυτό συμβαίνει διότι όσο περισσότερο αυξήσουμε τα βάρη των ακμών (μεγαλύτερες τιμές του L), τόσο θα ελαττώνεται η προτεραιότητά τους στην εκτέλεση του Kruskal, ενώ όταν τα ελαττώνουμε η προτεραιότητά τους αυξάνεται. Επομένως, όλοι οι δυνατοί βαθμοί για το S μπορούν να επιτευχθούν, ενώ το δέντρο παραμένει ΕΣΔ αφού τρέχουμε Kruskal. Παράλληλα, προκειμένου να διατηρήσουμε το ΕΣΔ και να αυξήσουμε ή να μειώσουμε τον βαθμό του S, θα πρέπει μία ακμή του E_S να γίνει ίση με κάποια άλλη ακμή του γράφου. Λαμβάνοντας όλες τις δυνατές διαφορές των ακμών με τον τρόπο που εξηγήθηκε παραπάνω, μπορούμε να μετατρέψουμε όλες τις ακμές του E_S ώστε να γίνουν κάποια στιγμή ίσες με κάθε άλλη ακμή του γράφου.

Πολυπλοκότητα

Ισχύει ότι: $O\left(\log(|E||E_S|)\right) \leq O\left(\log(|E|^2)\right) = O\left(2\log(|E|)\right) = O\left(\log(|E|)\right)$, οπότε:

$$\boxed{O\left(|E||E_S|\log(|E|) + |E|\log^2(|E|)\right)}$$

Άσκηση 2

Δίνεται γράφος $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{p})$

Παραδοχή: $e = (v, u)$ (από το v πας στο u) και $w(u)$, τότε $w(e) = w(u)$

1.

Αλγόριθμος & Ορθότητα

Παρατηρούμε πως το πρόβλημα ουσιαστικά ζητάει το μακρύτερο μονοπάτι από το s στο t, χωρίς να μηδενίζεται το συνολικό βάρος P στο ενδιάμεσο. Δεδομένου ότι δεν έχουμε κύκλους θετικών ακμών (επομένως δεν θα δημιουργηθούν κύκλοι αρνητικού μήκους κατά την μετατροπή), μπορούμε να δημιουργήσουμε το $-\boldsymbol{G}$, μετατρέποντας έτσι τις ελάχιστου βάρους ακμές σε μέγιστες και αντίστροφα. Τώρα, μπορούμε να τρέξουμε τον αλγόριθμο Bellman-Ford στο $-\boldsymbol{G}$ αναζητώντας το συντομότερο μονοπάτι (τροποποιώντας την βασική συνθήκη του από $D[u] > D[v] + w(v, u)$ σε $D[u] > D[v] + w(v, u) > 0$, ώστε να αποφύγουμε την ενημέρωση του $D[u]$ με αρνητικές αποστάσεις και να εξασφαλίσουμε ότι ο παίχτης δεν πεθαίνει). Τέλος, μετά το πέρας της επανάληψης αρκεί να ισχύει $D[t] < 0$ (αφού είμαστε στο $-\boldsymbol{G}$).

Πολυπλοκότητα

$$\boxed{O(|V||E|)}$$

2.

Αλγόριθμος & Ορθότητα

Τρέχουμε τον ίδιο αλγόριθμο με πριν $\big(O(|V||E|)\big)$. Αν μετά το πέρας του έχουμε $D[t] < 0$ τότε ο λαβύρινθος είναι r-ασφαλής. Αν τώρα έχουμε $D[t] > 0$ τότε ξανατρέχουμε άλλη μία (n-οστή) επανάληψη της επαναληπτικής μεθόδου του Bellman-Ford. Αν υπάρξει ενημέρωση σε κάποια τιμή του πίνακα D, τότε έχουμε κάποιον κύκλο αρνητικού μήκους (δηλαδή κύκλο θετικού μήκους στο $\boldsymbol{G}$), στον οποίο μπορούμε να καταλήξουμε ξεκινώντας από τον κόμβο s. Επομένως, αρκεί να δείξουμε ότι από κάποιο κόμβο του κύκλου αυτού καταλήγουμε στον κόμβο t (οπότε θα είναι r-ασφαλής για κάθε r). Εύκολα μπορούμε στην n-οστή επανάληψη να εντοπίσουμε τον κόμβο, του οποίου η απόσταση ενημερώθηκε, και να ακολουθήσουμε τους προγόνους του μέχρι να βρούμε όλες τις κορυφές του κύκλου (για όλους τους πιθανούς κύκλους). Τέλος, αγνοώντας πλέον τα βάρη,

μένει να δείξουμε ότι από μία οποιαδήποτε κορυφή από τις παραπάνω, έχουμε πρόσβαση στο t $\Big(DFS, BFS\ \big(O(|V| + |E|)\big)\Big)$.

Πολυπλοκότητα

$$\boxed{O(|V||E|)}$$

3.

Αλγόριθμος & Ορθότητα

Θέτουμε σαν r_{max} το άθροισμα όλων των θετικών βαρών των κορυφών, και $r_{min} = 1$. Στο διάστημα $[r_{min}, r_{max}] \subseteq \mathbb{Z}$ εκτελούμε δυαδική αναζήτηση $\Big(O\big(\log(r_{max})\big)\Big)$ και για κάθε r που λαμβάνουμε εφαρμόζουμε τον από πάνω αλγόριθμο $\big(O(|V||E|)\big)$ (ερώτημα 2.). Αν την πρώτη φορά που εκτελέσουμε Bellman-Ford βρούμε κύκλο θετικών βαρών (στο $\boldsymbol{G}$) που να ενώνεται με τις κορυφές s, t τότε η απάντηση είναι αυτομάτως $r = 1$.

Πολυπλοκότητα

$$\boxed{O\big(|V||E|\log(r_{max})\big)}$$

Άσκηση 3

Δίνεται γράφος $\boldsymbol{G(V, E)}$

Χωρίς βλάβη της γενικότητας υποθέτουμε ότι για κάθε $b(e)$ ισχύει: $b(e) \leq B$.

Αλγόριθμοι

1.

$cheapest[n][1]$ →προηγούμενοι φθηνότεροι

$cheapest[n][2]$ →επόμενοι φθηνότεροι

Για κάθε πόλη (σταθμό ανεφοδιασμού) v_i αποθηκεύουμε στην πρώτη γραμμή και στην στήλη i ενός πίνακα $2 \times n$ (έστω $cheapest[n][2]$ και δεικτοδότηση με αρχή το 1) τον δείκτη $m \in [2, n-1]$ της πόλης με την φθηνότερη βενζίνη, μεταξύ της v_i και των πόλεων που καταλήγουν στην v_i μέσω μιας ακμής $\big(O(n)\big)$. Με την ίδια λογική, αποθηκεύουμε στην δεύτερη γραμμή του πίνακα, στην θέση i, το δείκτη της πόλης με την φθηνότερη βενζίνη στην οποία μπορείς να πας από το v_i μέσω μιας ακμής. Σε περίπτωση ισοβαθμίας, επιλέγουμε τις πόλεις που βρίσκονται πιο κοντά στο t. Η διαδικασία αυτή γίνεται σε $O(n\log n)$ με χρήση priority queue που αποθηκεύει και ταξινομεί (ως προς το $c(i)$) δείκτες ενός συνόλου σταθμών οι οποίοι βρίσκονται μέσα σε ένα sliding window μήκους B. Διατρέχουμε τους n σταθμούς $\big(O(n)\big)$, προσθέτοντας κάποιον σταθμό i στην ουρά προτεραιότητας (βρίσκοντας έτσι το $cheapest[i][1]$) και ύστερα αφαιρώντας τον (βρίσκοντας έτσι το $cheapest[i][2]$) $\big(O(\log n)\big)$. Ακολούθως, εντοπίζουμε τους σταθμούς v_p για τους οποίους ισχύει $cheapest[p][1] = p$ (τους ονομάζουμε critical points) $\big(O(n)\big)$. Τέλος, για να πάμε από το ένα critical point στο επόμενο αρχίζοντας και τελειώνοντας με μηδενική βενζίνη, ακολουθούμε την εξής διαδικασία για κάθε critical point p:

Αν $b(\{p, p+1\}) \leq B$, τότε γέμισε όση βενζίνη υπολείπεται και πήγαινε στο $p+1$,

Αλλιώς γέμισε το ντεπόζιτο και μετακινήσου στο $cheapest[p][2]$ (επόμενο φθηνότερο). Θέσε $p = cheapest[p][2]$ και επανέλαβε μέχρι να πάμε στο επόμενο critical point.

2.

Θα χρησιμοποιήσουμε δυναμικό προγραμματισμό $\big(O(n^2)\big)$. Ορίζουμε αρχικά ένα σύνολο $GAS(x) = \{B - b(\{k, x\}) \mid k \in V \text{ και } c(k) < c(x) \text{ και } b(\{k, x\}) \leq B\} \cup \{0\}$

προκειμένου να κρατάμε όλες τις πιθανές τιμές της βενζίνης για κάθε x.

Ορίζουμε τώρα το ελάχιστο κόστος να πάμε από έναν κόμβο (πόλη-σταθμό) $x \neq t$ ($L(t,0)=0$) στον t, για $g \in GAS(x)$ ως:

$$L(x,g) = min\begin{Bmatrix} L(k,0) + (b(\{k,x\}) - g)c(x) & : c(k) \leq c(x) \wedge g \leq b(\{k,x\}) \\ L\big(k, B - b(\{k,x\})\big) + (B-g)c(x) & : c(k) > c(x) \end{Bmatrix},$$

$$b(\{k,x\}) \leq B$$

Αφού υπολογίσουμε τις τιμές του L, το πρόβλημα ανάγεται στον υπολογισμό συντομότερου μονοπατιού για ένα καινούριο κατευθυνόμενο γράφο $\boldsymbol{G'(V', E')}$ με θετικά βάρη. Με άλλα λόγια το $L(x,g)$ θα ισούται με το μήκος του συντομότερου μονοπατιού από το (x,g) στο $(t,0)$ στο $\boldsymbol{G'}$**.**

Ο γράφος αυτός θα έχει την ακόλουθη μορφή:

Κάθε κορυφή θα είναι της μορφής (x,g)

Οι ακμές του γράφου τα βάρη τους θα προκύπτουν από τους υπολογισμούς του L, έτσι ώστε για κάθε $x,k \in V$ και $g \in GAS$, όπου $b(\{k,x\}) \leq B$, θα ισχύει $w\big((x,g),(k,0)\big) = (b(\{k,x\}) - g)c(x)$, για $c(k) \leq c(x) \wedge g \leq b(\{k,x\})$ ή $w\left((x,g),\big(k, B - b(\{k,x\})\big)\right) = (B-g)c(x)$, για $c(k) > c(x)$.

Θα ισχύει επομένως $|\boldsymbol{V'}| = n \cdot B$ το πολύ.

Τέλος, τρέχουμε τον αλγόριθμο Dijstra και βρίσκουμε το συντομότερο μονοπάτι $\big(O((n \cdot B)^2)\big)$.

Ορθότητα

1.

Πρόταση 1

Θέλουμε να βρούμε έναν τρόπο να επιλέγουμε την ποσότητα βενζίνης που ανεφοδιάζουμε. Έστω $stops = \{v_1, v_2, \dots, v_k\}$ το σύνολο των στάσεων ανεφοδιασμού της βέλτιστης λύσης. Προφανώς, όταν θα έχουμε φτάσει στο v_k θα γεμίσουμε το ντεπόζιτο μας με την ελάχιστη δυνατή βενζίνη ώστε να φτάσουμε στο t με $B=0$. Για όλες τις ενδιάμεσες στάσεις θα εφαρμόσουμε την ακόλουθη λογική:

Αν $c(v_i) < c(v_{i+1})$, τότε γεμίζουμε το ντεπόζιτο στην v_i, **αλλιώς** γεμίζουμε μόνο όσο υπολείπεται από το ντεπόζιτο (μέχρι να ισχύει $b(\{v_i, v_{i+1}\}) = B$) για να φτάσουμε στην v_{i+1}.

Απόδειξη

Έστω ότι $c(v_i) < c(v_{i+1})$ και ότι η βέλτιστη λύση δεν ανεφοδιάζεται στο v_i. Τότε αν ανεφοδιαστούμε με περισσότερη βενζίνη στο v_i και με λιγότερη στο v_{i+1} (κατά την ίδια ποσότητα), είναι προφανές πως θα πετυχαίναμε καλύτερη λύση από ότι προηγουμένως, γεγονός που αναιρεί την υπόθεση της βέλτιστης λύσης. Όμοια ισχύει και για την περίπτωση $c(v_i) \geq c(v_{i+1})$.

Πρόταση 2

Μπορούμε να βρούμε βέλτιστη λύση για το πρόβλημα για την οποία καταλήγουμε σε κάθε πόλη v_p, όπου $cheapest[p][1] = p$ (critical point), με μηδενική βενζίνη.

Απόδειξη

Έστω $stops = \{v_1, v_2, \dots, v_k\}$ το σύνολο των στάσεων ανεφοδιασμού της βέλτιστης λύσης. Αφού $cheapest[i][1] = i$ τότε σημαίνει πως $c(v_i) \leq c(v_{i-1})$, οπότε σύμφωνα με την πρόταση 1, στο v_{i-1} θα γεμίζουμε την ελάχιστη δυνατή βενζίνη που χρειαζόμαστε για να φτάσουμε στην v_i. Άρα, θα καταλήγουμε με άδειο ντεπόζιτο στην v_i.

Πολυπλοκότητα

1. $O(n\log n)$

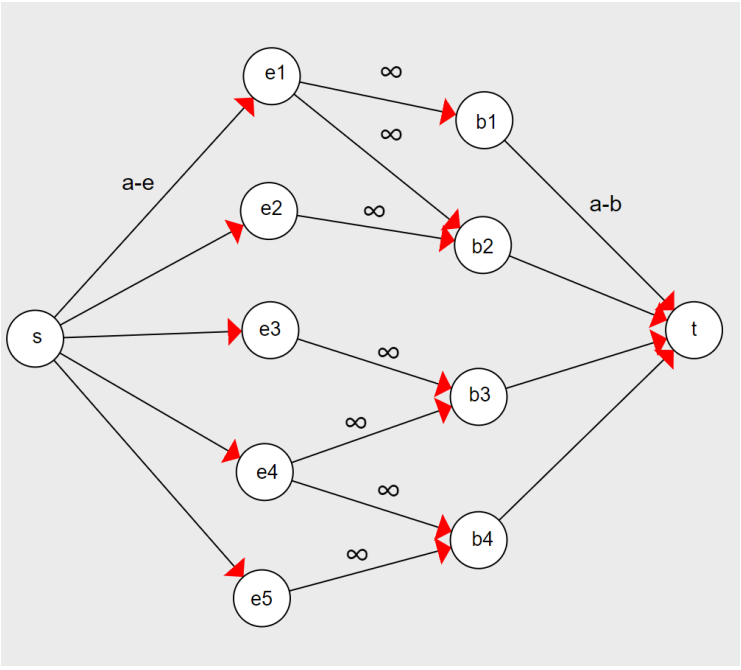
2. $O((n \cdot B)^2)$

Άσκηση 4

Έχουμε πίνακες: $A = \{a_1, a_2, \dots, a_n\}, B = \{b_1, b_2, \dots, b_k\}, E = \{e_1, e_2, \dots, e_m\}$

Αλγόριθμος

Αρχικά ταξινομούμε τους πίνακες A, B, E (εφόσον δεν είναι εξαρχής) $\big(O(n\log n + k\log k + m\log m)\big)$. Ύστερα, υπολογίζουμε την ελάχιστη απόσταση κάθε βάσης και κάθε εξτρεμιστή από τον κοντινότερο στρατιώτη. Αυτό γίνεται εύκολα σε γραμμικό χρόνο διατρέχοντας τους 3 πίνακες $\big(O(k+m)\big)$, μιας και οι πίνακες είναι ταξινομημένοι (μπορεί να μην χρειαστεί να διατρέξουμε όλον τον Α), δημιουργώντας έτσι τις ελάχιστες αποστάσεις $|a - \mathrm{b}|$ και $|a - e|$, $\forall e \in E, \forall b \in B$ και $a \in A$. Ακολούθως, σχηματίζουμε με τα δεδομένα αυτά ένα (διμερές) κατευθυνόμενο γράφημα (δίκτυο), όπου κάθε $e \in E$ ενώνεται με έναν κόμβο s (ροή $s \to e$) και κάθε $b \in B$ ενώνεται με έναν κόμβο t (ροή $b \to t$). Ως χωρητικότητες ακμών, για κάθε $s \to e$ βάζουμε την αντίστοιχη απόσταση $a - e$, και όμοια για κάθε $b \to t$ βάζουμε την αντίστοιχη απόσταση $a - b$. Στην συνέχεια, υπολογίζουμε όλες τις αποστάσεις $|e - b| \le d$ $(O(mk))$ και δημιουργούμε μία αντίστοιχη γραμμή στο γράφημα με άπειρη χωρητικότητα. Τελικά το γράφημα θα αποκτήσει την μορφή που φαίνεται ενδεικτικά παρακάτω. Τέλος, αρκεί να εκτελέσουμε τον αλγόριθμο Ford–Fulkerson $\Big(O\big((m+k+2)(m+k+mk)^2\big)\Big)$ και να υπολογίσουμε την συνολική ροή στον δίκτυο.



Ορθότητα

Μία προφανή παρατήρηση είναι το γεγονός ότι προκειμένου ο στρατός να καταστρέψει μία βάση ή να αιχμαλωτίσει δυνάμεις εξτρεμιστών, αρκεί να στείλει τις κοντινότερες δυνάμεις στρατού σε εκείνη την θέση. Μιας και ο στρατός ξαναγυρνάει στην αρχική του θέση μετά την ενέργεια αυτή, οι σχετικές θέσεις στρατού, εξτρεμιστών και βάσεων παραμένουν σταθερές. Επομένως, η υπόθεση αυτή θα ισχύει πάντα. Συνεπώς, θέτουμε τις ακμές από κάθε e_i και b_i προς τα s και t αντίστοιχα, με βάση τις κοντινότερες αποστάσεις με κάποια δύναμη στρατού. Επιπλέον, θέτοντας άπειρη τιμή ροής μεταξύ βάσεων και εξτρεμιστών (ακμές όπου $|b_i - e_i| \leq d$) , εξασφαλίζουμε ότι κατά τον υπολογισμό του Max-Flow δεν θα έχουμε περιορισμούς ροής, οπότε θα υπολογίζεται ολόκληρη η ροή που διέρχεται από μία ακμή $a_i - b_i$ ή $a_i - e_i$ για κάθε μονοπάτι $s - t$. Διαισθητικά καταλαβαίνουμε πως αρκεί απλά να συναθροίσουμε όλες τις ακμές $a_i - b_i$ και $a_i - e_i$ στις οποίες έχουμε μέγιστη ροή (αν $a_i - b_i = a_i - e_i$ τότε λαμβάνουμε υπόψιν την απόσταση μία φορά). Με άλλα λόγια αρκεί να βρούμε την συνολική ροή που εξέρχεται από το s (ή που καταλήγει στο t).

Πολυπλοκότητα

$$O\big((m+k+2)(m+k+mk)^2\big)$$

Άσκηση 5

Τακτοποίηση Ορθογωνίων Παραλληλογράμμων

Θα δείξουμε ότι $Partition \leq_P ΤΟΠ$.

Έστω n ορθογώνια παραλληλόγραμμα $A_1, \dots, A_n$, διαστάσεων $\varepsilon \times x_1, \dots, \varepsilon \times x_n$, με $\varepsilon \ll 1$.

Έστω ακόμα ορθογώνιο παραλληλόγραμμο B, διαστάσεων $2\varepsilon \times s$. Θέτουμε ακόμα:

$$s = \frac{1}{2}\sum_{i=1}^{n} x_i$$

Ως εκ τούτου, το να χωρίσουμε το σύνολο $\{x_1, \dots, x_n\}$ σε δύο ίσα (αθροιστικά) σύνολα, είναι το ίδιο πράγμα με το να τοποθετήσουμε τα $A_1, \dots, A_n$ μέσα στο B, μιας και το πλάτος του είναι διπλάσιο από το πλάτος των $A_1, \dots, A_n$. Πρέπει επομένως να επιλέξουμε κατάλληλο ε ώστε να αποτρέψουμε πιθανές περιστροφές των $A_1, \dots, A_n$.

Μέγιστη τομή με Βάρη στις Κορυφές

Θα δείξουμε ότι $Partition \leq_P MTBK$.

Έστω, αρχικά, μια διαμέριση του γράφου $(S, V/S)$, με τομή βάρους $w(S, V/S)$. Θέλουμε $w(S, V\backslash S) \geq B$. Επειδή έχουμε πλήρες γράφημα θα ισχύει:

$$w(S, V\backslash S) = \sum_{w_v \in S, w_u \in (S, V/S)} w_v w_u = \sum_{w_v \in S} w_v \sum_{w_u \in (S, V/S)} w_u \geq B$$

Παρατηρούμε ότι το πρόβλημα είναι πολυωνυμικά ισοδύναμο με το πρόβλημα Maximum-Cut.

Θέτουμε ακόμα

$$W = \sum_{w_i, i \in V} w_i$$

δηλαδή το συνολικό άθροισμα βαρών όλων των ακμών και προκύπτει:

$$w(S, V/S) = w(S)\big(W - w(S)\big)$$

Λαμβάνοντας υπόψιν την κοίλη συνάρτηση $f(x) = x(W - x)$, η οποία παρουσιάζει μέγιστο στο $x = \frac{W}{2}$ θα έχουμε:

$$w(S)\big(W - w(S)\big) \leq \frac{W^2}{4}$$

Από τα παραπάνω παρατηρούμε ότι το $Partition$ μπορεί να αναχθεί στο δικό μας πρόβλημα. Πιο συγκεκριμένα, αν είχαμε ένα σύνολο V θετικών ακεραίων και θέλαμε να το διαμερίσουμε σε δύο ισοδύναμα σύνολα $S, V/S$, θα μπορούσαμε να δημιουργήσουμε ένα γράφο $G(V, E)$ (με βάρη στις κορυφές όσο και οι τιμές των ακεραίων του συνόλου V) και να μεγιστοποιήσουμε την τομή του.